Load-Balance & Proxies

Networking

Pierre-Philipp Braun <pbraun@nethence.com>

Table of contents

Scalability

First, how to be DoS resilient?…

==> infrastructure *and* code done right

- Principle of least privilege
  - firewall policy
  - cluster services need to listen internally only
  - service/app authentication & authorization
- KISS / no hype required, less code == less vulns == less bugs
- Code harden & pay devs to REMOVE lines
- Fast incident response (about monitoring and well-established HA processes, not about forensics)

Second, how to be high-load resilient?…

==> load-balance ready

- Load-balance cluster – apps are stateless and/or cluster-aware
- Distributing the network load is the true active/active
- Shared data storage
- Big-enough pipes

```
      X                               XX
      XX         public network        X
                                        X
         XXX XXX                      XX
                  X X X  X   XX   X X

  9.9.9.9
+---------+      +---------+      +---------+
|         |      |         |      |         |
|  node1  |      |  node2  |      |  node3  |
|         |      |         |      |         |
+---------+      +---------+      +---------+
```

- got Swarm or K8S cluster
- one application end-point with public IP `9.9.9.9`
- DNS record `app.example.net IN A 9.9.9.9`

Everything is fine there?

==> NO – all the load goes to only one node

- Swarm and K8S do have a network overlay by default, which re-distributes load to other nodes
- however node1 becomes a load-balancer *ipso-facto* here
- node1 is not necessarily sized for that purpose

How to design `app.example.net` *so the traffic gets shuffled around the nodes?...*

==> TWO SOLUTIONS

- either by means of DNS round-robin
- –or– by means of a load-balancer in between

```
      X                              XX
     XX                              X
                public network        X
        XXX XXX                     XX
                 X X X  X  XX   X X

           +---------------------+
           |                     |
    +------+    LOAD BALANCE     +-------+
    |      |                     |       |
    |      +-----------+---------+       |
    |                  |                 |
    |                  |                 |
+---+-----+      +-----+---+      +------+--+
|         |      |         |      |         |
|  node1  |      |  node2  |      |  node3  |
|         |      |         |      |         |
+---------+      +---------+      +---------+
```

Balancing methods

- Layer 3 round-robin
- Layer 3 round-robin **with “sticky connection” –** *reminds src/internal-dst IP addresses*
- Layer 7

Commercial load-balancers

not sure what part is truly hardware-based

- F5 BigIP
- Fortinet FortiGate
- …

Open Source load-balancers

- layer-3 BSD `pf` vs. `npf` vs. `ipfilter`/`ipnat`
- layer-3 Linux Netfilter (`iptables` vs. `nft`)
- layer-3 eBPF // LAB PoC that!
- layer-3+7 HAProxy
- layer 7 NGINX / NGINX Plus (dynamic objects?)
- layer 7 Apache Traffic Server? (static objects?)
- layer 7 OpenBSD Relayd
- K8S Ingress / Ingress-NGINX

So can we just replace the previously seen HA setups with load-balancing?

May we simply forget the tradition?…

load-balancing != HA

==> Yes and No

- Yes as long as orchestrator manages the instances **and VIPs**
- No if nothing takes care of the cluster health already

HA-capable load-balance

Which ones are HA capable against the back-end nodes/services?

==> With additional scripts, probably any – though maybe not that corporate nor resilient

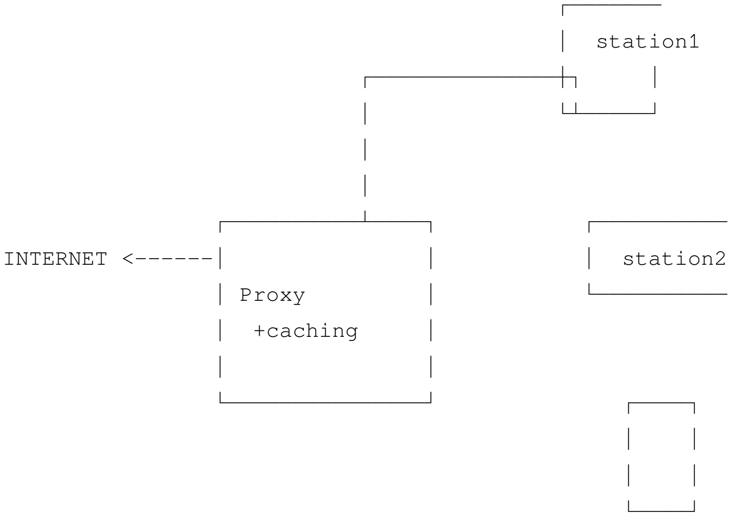
==> Built-in for sure *aka* **Health Checks**

checking the back-end service

- layer-3+7 HAProxy
- layer 7 NGINX / NGINX Plus
- layer 7 OpenBSD Relayd? // LAB PoC that!

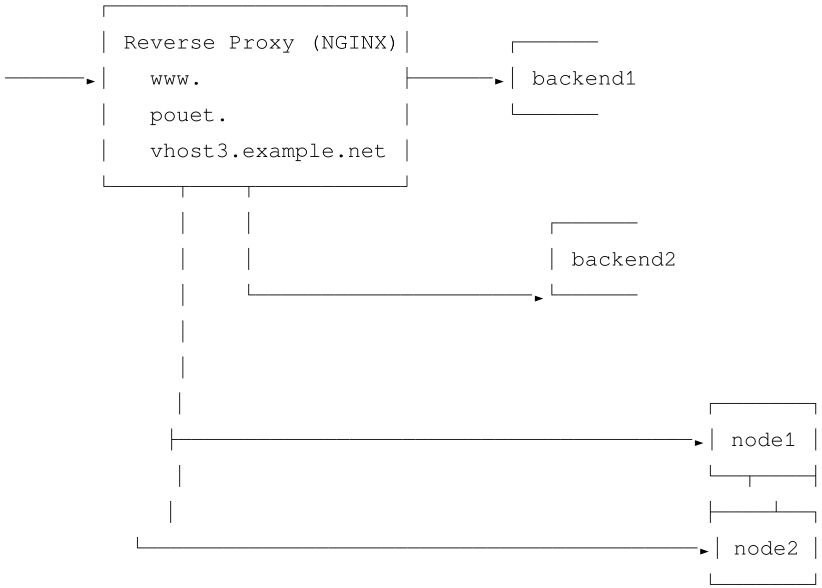
Everybody clear on what is a Proxy vs. Reverse Proxy?…

Proxy (outbound) - restrict access & caching



```
export HTTP_PROXY=http://10.1.1.252:8080
curl ...
wget ...
```

Reverse-proxy (inbound) - SSL & HTTP termination



```
vi /etc/nginx/conf.d/vhost.conf

server {
    ...

        location / {
                proxy_pass http://APPLICATION_ADDRESS:PORT;
        }
}
```

think of fault-tolerance again

What was missing from the diagram above?…

Redundant load-balance

==> THERE WAS A SPOF!

The balancers need to be fault-tolerant also

ACTIVE/PASSIVE - NO DNS ROUND ROBIN

```
      X                               XX
     XX                               X
                public network         X
        XXX XXX                      XX
                 X X X  X   XX   X X

   +----------------+   +----------------+
   |                |   |                |
   +  LOAD BALANCE  +---+  LOAD BALANCE  +
   |                |   |                |
   +----------------+   +----------------+

+---------+      +---------+      +---------+
|         |      |         |      |         |
|  node1  |      |  node2  |      |  node3  |
|         |      |         |      |         |
+---------+      +---------+      +---------+
```

Would you consider ACTIVE/ACTIVE + DNS ROUND ROBIN?

How to make it possible?…

ACTIVE/ACTIVE + DNS ROUND ROBIN

==> only if you got TWO VIPs (ideally CARP/VRRP-based)

Be it active/passive or active/active

Which balancers can be made fault-tolerant themselves?…

==>

- `pf` with `pfsync`
- any other filter as long as you keep the rules in sync // LAB what about the states?…
- and enable some IP failover(s) – VRRP/CARP is better than a floating IP

LAB OR BONUS // can L7 lbs be made fault-tolerant themselves?

when it is more than just high load to handle

How to resist DDoS attacks?…

==> DDoS resilient

DDoS Protection at some NOC & ISP

- Arbor® Networks Peakflow
- Sevi® M6-NG

CDN - *a world-wide load-balance scenario*

- got more reverse-proxies than backends
- your backend is unknown to the end-users
- your DMZ is somehow the public network itself

BGP 666 black-hole at some NOC

- ISP only looses one customer

// Questions on scalability?

Load-Balance++

What was the trick again to handle L7 session at L3?…

==> OpenBSD's Packet Filter "sticky connection"

```
match in on egress proto tcp to port 80
    rdr-to 10.0.0.10, 10.0.0.11, 10.0.0.13
    round-robin sticky-address
```

LAB // find out and PoC a scenario where PF's "sticky connection" feature would help

Load-balance algorithms

Network architectures

- DNAT - ok performance
- FULLNAT - less-ok performance
- Direct Routing - best performance

The DNAT use-case

- balancer(**s**) on Perimeter and specific internal VLAN
- dedicated little cluster DMZ just for it
- so internal traffic can remain clear-text
- (we're looking for performance here)

L7-only

- internal connections possibly ~pooled with HTTP keepalive // LAB here

DNAT vs. FULLNAT

DNAT

```
inbound packet: change dst
```

FULLNAT (according to their diagram, this is not outbound SNAT)

```
inbound packet: change src/dst
```

LAB // dig into FULLNAT – what are its advantages? Is it just about avoiding SNAT and internet access to the farm?

Direct Routing *(some kind of an L2 trickery)*

aka Direct Server Return

aka nPath routing

aka IBM IPVS aka LVS…

- everybody’s on Perimeter!
- only inbound traffig gets shuffled
- nodes share a **Virtual IP**
- VIP does not respond to ARP requests **but can receive traffic**
- outbound traffic goes as nodes’ real IP

Pure HWLB products?

not sure it's ASIC nor true offloading based even

- F5 BigIP
- Fortinet FortiGate
- (Citrix Netscaler)
- (no Juniper nor Cisco anymore)
- Pulse Secure vADC (formerly Zeus)

Not-so-HWLB

just LVS + HAProxy + Linux?

Small & medium-sized

- Barracuda Networks
- Loadbalancer.org
- Kemp Technologies

Hardware vs. software

- traditionally L3/L4
- now some do SSL termination
- and some are even L7

LAB // try to reproduce an L3/L4 lb issue against L7 streaming/keep-alive connections

L3 r0cks!

- performance = the lower the layer you can get
- reliable = the simpler you can get
- *and the sticky connection trick*

Now what are the advantages of L7 over L3?...

They say L7 rocks

- SSL-offloading *for sure*
- possibly optimize the content (compression, etc.)
- buffers and offloads slow connections from the upstream servers
- possibly throttling/postscreen (slow down zombies)

LAB // HTTP compression (not SSL) vuln to leverage within SSL?

L3 vs. L7

L3

- issues with user cookies handled on a single backend instance
- issues with streaming/keep-alive connections?

L7 – *ask a modern software/application architect*

- better session handling against containers & microservices?
- terminates the HTTP session
- configuration flexibility e.g. NGINX tweaks and redirects
- can make load-balance decision based on requested URL, HTTP headers, user cookie, …
- possibly Web Application Firewall (WAF) capable

Heavy-load capable

- backend instances may not handle high-load by design
- L7 reverse-proxies designed to handle high-load
- NGINX vs. Apache vs. even worse
- Thttpd & others are also good to serve just images & static pages

lb.org features

round-robin algorithms
    Scheduling and balancing methods

```
Round Robin
Weighted Round Robin
Least Connection
Weighted Least Connection
Agent-based Adaptive (Windows and Linux Agents)
Layer 7 Content Switching
Destination Hash for transparent proxy
```

LAB // other balancing methods than RR for linux/bsd?

L4/L7 sticky persistence

Source IP address

SSL Session ID

Passive Cookie

Active Cookie (Insert)

RDP Cookie/Session Broker

X-forwarded for header (better than Super HTTPS)

Port following (Persistence on multiple combined ports)

Multiple fallback options i.e. use source IP if no cookie found

Health checking and high availability

Application health checking for DNS, FTP, HTTP, HTTPS, IMAP, NNTP and ma

ICMP health checking of server farm machines

Complex manually scripted health checks

Automatic reconfiguration for defective real server machines

Automatically remove a failed server from the load balancing pool

Automatic replication of static and dynamic configuration from master t

Stateful Failover (persistence table replication)

One click secure clustered pair configuration

// Questions on load-balancers?

Forward & Reverse Proxies

all kinds of proxies

Goal is to identify best products for

- L7 load-balance reverse-proxy
- CDN reverse-proxy with advanced caching
- forward-proxy with caching

In the old days…

- `HTTP_PROXY` variable meant something
- otherwise in FF & Chrome settings

What happened? Why did this practice mostly disapear?…

The SSL situation

==> HTTPS is end-to-end encryption **with authentication**

we will discuss this goal later-down

General proxy types

Routing pattern

- Forward proxies
- Reverse proxies
- Tunneling proxies

Access protocol

- HTTP Proxy
- FTP Proxy
- SSL Proxy
- SOCKS Proxy

Kinds of proxies…

- reverse vs. forward
- no caching vs. basic caching vs. advanced caching

What would be the features required to build up a CDN?…

==> reverse + advanced caching

Basic caching functionality

- NGINX – *supposedly reverse…* // LAB forward
- HAProxy – *reverse-only?* // LAB forward?
- ATS – *reverse & possibly forward* // LAB (see resources)
- Envoy w/ eCache – *welcome to devops style*
- Polipo – *HTTP only (SSL terminate elsewhere)*

Caching horses

- Squid – *forward & possibly reverse*
- Varnish – *super cool reverse-only* // BONUS-OR-LAB why not forward?
- Nuster (HAProxy-based) – *reverse-only?* // LAB forward?
- WWWOFFLE – *forward & dialup-ready*

Don’t worry be happy

LAB // identify, differenciate and compare caching features and techniques

LAB // benchmark those proxies – which one is best? e.g. NGINX vs. HAProxy

LAB // NGINX Health Checks without the “Plus”?

LAB // NGINX & ATS caching & fwd caching

// Questions on forward and reverse proxies?

More on the SSL situation

We got stuck with HTTPS.

- certificate gets verified by browser
- and eve-proof symmetric key negociation

How to solve that?...

==> MITM-by-design required

Transparent

- SSL-capable forward proxy to intercept
- SSL termination from the inside & sign on the fly

LAB // possibly non-transparent? Does it solve the on-the-fly problem?

Corporate HTTP_PROXY for the 21th century

We could secure an internal network as such

- internet access goes only through the corporate proxy
- optionally check client certificates to grant access to it

Would require

- deploy CA certificate on workstations
- deploy client certs on workstations

Not bullet proof

- SSL VPN tunnels still went through
- and would still go through with this design

LAB // What about `HTTP_PROXY=https://internal-proxy:8443`?

LAB // Otherwise PoC MITM on-the-fly with a caching capable fwd proxy.

LAB // is there a way to block SSL VPN tunnels? (hint: IDS/IPS & DPI)

This is the end # More lab opportunities

Load Balance Outgoing Traffic
https://www.openbsd.org/faq/pf/pools.html#outgoing